Appears in
Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000),
Austin, TX, July 2000.

Local search characteristics of incomplete SAT procedures

Dale Schuurmans and **Finnegan Southey**
Department of Computer Science
University of Waterloo
{dale,fdjsouth}@cs.uwaterloo.ca

Abstract

Effective local search methods for finding satisfying assignments of CNF formulae exhibit several systematic characteristics in their search. We identify a series of measurable characteristics of local search behavior that are predictive of problem solving efficiency. These measures are shown to be useful for diagnosing inefficiencies in given search procedures, tuning parameters, and predicting the value of innovations to existing strategies. We then introduce a new local search method, SDF ("smoothed descent and flood"), that builds upon the intuitions gained by our study. SDF works by greedily descending in an informative objective (that considers how strongly clauses are satisfied, in addition to counting the number of unsatisfied clauses) and, once trapped in a local minima, "floods" this minima by re-weighting unsatisfied clauses to create a new descent direction. The resulting procedure exhibits superior local search characteristics under our measures. We then show that our method is competitive with the state of the art techniques, and typically reduces the number of search steps by a significant factor.

Introduction

Since the introduction of GSAT (Selman, Levesque, & Mitchell 1992) there has been considerable research on local search methods for finding satisfying assignments for CNF formulae. These methods are surprisingly effective; they can often find satisfying assignments for large CNF formulae that are far beyond the capability of current systematic search methods (however see (Bayardo & Schrag 1997; Li & Anbulagan 1997) for competitive systematic search results). Of course, local search is incomplete and cannot prove that a formula has no satisfying assignment when none exists. However, despite this limitation, incomplete methods for solving large satisfiability problems are proving their worth in applications ranging from planning to circuit design and diagnosis (Selman, Kautz, & McAllester 1997; Kautz & Selman 1996; Larrabee 1992).

Significant progress has been made on improving the speed of these methods since the development of GSAT. In fact, a series of innovations have led to current search methods that are now an order of magnitude faster.

Perhaps the most significant early improvement was to incorporate a "random walk" component where variables were flipped from within random falsified clauses (Selman & Kautz 1993). This greatly accelerated search and led to the development of the very successful WSAT procedure (Selman, Kautz, & Cohen 1994). A contemporary idea was to keep a tabu list (Mazure, Saïs, & Grégoire 1997) or break ties in favor of least recently flipped variables (Gent & Walsh 1993; 1995) to prevent GSAT from repeating earlier moves. The resulting TSAT and HSAT procedures were also improvements over GSAT, but to a lesser extent. The culmination of these ideas was the development of the Novelty and R_Novelty procedures which combined a preference for least recently flipped variables in a WSAT-type random walk (McAllester, Selman, & Kautz 1997), yielding methods that are currently among the fastest known.

A different line of research has considered adding clause-weights to the basic GSAT objective (which merely counts the number of unsatisfied clauses) in an attempt to guide the search from local basins of attraction to other parts of the search space (Frank 1997; 1996; Cha & Iwama 1996; 1995; Morris 1993; Selman & Kautz 1993). These methods have proved harder to control than the above techniques, and it has only been recent that clause re-weighting has been developed to a state of the art method. The series of "discrete Lagrange multiplier" (DLM) systems developed in (Wu & Wah 1999; Shang & Wah 1998) have demonstrated competitive results on benchmark challenge problems in the DIMACS and SATLIB repositories.

Although these developments are impressive, a systematic understanding of local search methods for satisfiability problems remains elusive. Research in this area has been largely empirical and it is still often hard to predict the effects of a minor change in a procedure, even when this results in dramatic differences in search times.

In this paper we identify three simple, intuitive measures of local search effectiveness: depth, mobility, and coverage. We show that effective local search methods for finding satisfying assignments exhibit all three characteristics. These, however, are conflicting demands and successful methods are primarily characterized by their ability to effectively manage the tradeoff between these factors (whereas ineffective methods tend to fail on at least one measure). Our goal is to be able to distinguish between effective and ineffective search strategies in a given problem (or diagnose problems with a given method, or tune parameters) without having to run exhaustive search experiments to their completion.

To further justify our endeavor, we introduce a new local search procedure, SDF ("smoothed descent and flood") that arose from our investigation of the characteristics of effective local search procedures. We show that SDF exhibits uniformly good depth, mobility, and coverage values, and consequently achieves good performance on a large collection of benchmark SAT problems.

Local search procedures

In this paper we investigate several dominant local search procedures from the literature. Although many of these strategies appear to be only superficial variants of one another, they demonstrate dramatically different problem solving performance and (as we will see) they exhibit distinct local search characteristics as well.

The local search procedures we consider start with a random variable assignment $\mathbf{x} = \langle x_1, ..., x_n \rangle$, $x \in \{0, 1\}$ and make local moves by flipping one variable $x'_i = 1 - x_i$ at a time, until they either find a satisfying assignment or time out. For any variable assignment there are a total of n possible variables to consider, and the various strategies differ in how they make this choice. Current methods uniformly adopt the original GSAT objective of minimizing the number of unsatisfied clauses, perhaps with some minor variant such as introducing clause weights or considering how many new clauses become unsatisfied by a flip (break count) or how many new clauses become satisfied (make count). The specific flip selection strategies we investigate (along with their free parameters) are as follows.

GSAT$()$ Flip the variable x_i that results in the fewest total number of clauses being unsatisfied. Break ties randomly. (Selman, Levesque, & Mitchell 1992)

HSAT$()$ Same as GSAT, but break ties in favor of the least recently flipped variable. (Gent & Walsh 1993)

WSAT-G(p) Pick a random unsatisfied clause c. With probability p flip a random x_i in c. Otherwise flip the variable in c that results in the smallest total number of unsatisfied clauses. (McAllester, Selman, & Kautz 1997)

WSAT-B(p) Like WSAT-G except, in the latter case, flip the variable that would cause the smallest number of new clauses to become unsatisfied. (McAllester, Selman, & Kautz 1997)

WSAT(p) Like WSAT-B except first check whether some variable x_i would not falsify any new clauses if flipped, and always take such a move if available. (Selman, Kautz, & Cohen 1994; McAllester, Selman, & Kautz 1997)

Novelty(p) Pick a random clause c. Flip the variable x_i in c that would result in the smallest total number of unsatisfied clauses, unless x_i is the most recently flipped variable in c. In the latter case, flip x_i with probability $1 - p$ and otherwise flip the variable x_j in c that results in the second smallest total number of unsatisfied clauses. (McAllester, Selman, & Kautz 1997)

Novelty+(p, h) Like Novelty, except that after the clause c is selected, flip a random x_i in c with probability h, otherwise continue with Novelty. (Hoos 1999)

Note that, conventionally, these local search procedures have an outer loop that places an upper bound, F, on the maximum number of flips allowed before re-starting with a new random assignment. However, we will not focus on random restarts in our experiments below because *any* search strategy can be improved (or at the very least, not damaged) by choosing an appropriate cutoff value F (Gomes, Selman, & Kautz 1998). In fact, it is straightforward and well known how to do this optimally (in principle): For a given search strategy and problem, let the random variable f denote the number of flips needed to reach a solution in a single run, and let f_F denote the number of flips needed when using a random restart after every F flips. Then we have the straightforward equality (Parkes & Walser 1996)

$$\mathrm{E} f_F \;=\; F/\mathrm{P}(f \le F) - [F - \mathrm{E}(f | f \le F)]$$

Note that this always offers a potential improvement since

$$\begin{aligned} \mathrm{E} f_F &= F\mathrm{P}(f > F)/\mathrm{P}(f \le F) + \mathrm{E}(f | f \le F) \\ &\le F\mathrm{P}(f > F) + \mathrm{E}(f | f \le F) \quad \le \quad \mathrm{E} f \end{aligned}$$

for any cutoff $F > 0$. In particular, one could choose the optimal cutoff value $F^* = \arg\min_F \mathrm{E} f_F$. We report this optimal achievable performance quantity for every procedure below, using the empirical distribution of f over several runs to estimate $\mathrm{E} f_{F^*}$. Thus we will focus on investigating the single run characteristics of the various variable selection policies, but be sure to report estimates of what the optimum achievable performance would be using random restarts.

Measuring local search performance

In order to tune the parameters of a search strategy, determine whether a strategic innovation is helpful, or even debug an implementation, it would be useful to be able to measure how well a search is progressing without having to run it to completion on large, difficult problems.

To begin, we consider a simple and obvious measure of local search performance that has no doubt been used to tune and debug many search strategies in the past.

Depth measures how many clauses remain unsatisfied as the search proceeds. Intuitively, this indicates how deep in the objective the search is remaining. To get an overall summary, we take a depth average over all search steps. Note that it is desirable to obtain a small value of depth.

Although simple minded, and certainly not the complete story, it is clear that effective search strategies do tend to descend rapidly in the objective function and remain at good objective values as the search proceeds. By contrast, strategies that fail to persistently stay at good objective values usually have very little chance of finding a satisfying assignment in a reasonable number of flips (McAllester, Selman, & Kautz 1997).

To demonstrate this rather obvious effect, consider the problem of tuning the noise parameter p for the WSAT procedure on a given problem. Here we use the uf100-0953

100 runs on uf100-0953	Avg. depth	Avg. flips	Est. opt. w/cutoff
WSAT$(.5)$	5.62	11,153	9,618
WSAT$(.7)$	8.65	16,772	14,926
WSAT$(.8)$	10.3	29,322	25,835
WSAT$(.9)$	12.1	48,175	48,175
Novelty$(.5)$	4.85	3,958	3,958
SDF$(\frac{2}{m}, .995)$	3.93	1,242	1,242

Figure 1: Depth results

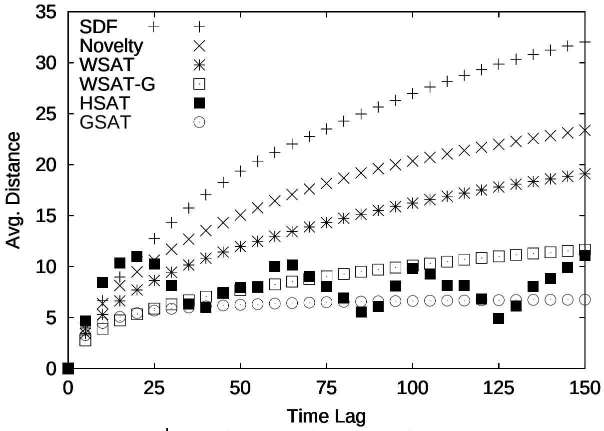
Flips	Depth rank				Mobility rank			
rank	best 1	2	3	worst 4	best 1	2	3	worst 4
best 1	.82	.09	.05	.04	.81	.11	.07	.01
2	.08	.39	.30	.16	.11	.50	.30	.08
3	.07	.36	.41	.23	.07	.25	.50	.18
worst 4	.03	.16	.25	.57	.01	.13	.13	.73

Figure 2: Large scale experiments (2700 uf problems)

problem from the SATLIB repository to demonstrate our point.[1] Figure 1 shows that higher noise levels cause WSAT to stay higher in the objective function and significantly increase the numbers of flips needed to reach a solution. This result holds both for the raw average number of flips but also for the optimal expected number of flips using a maximum flips cutoff with random restarts, $\hat{\mathrm{E}}f_{F^*}$. The explanation is obvious: by repeatedly flipping a random variable in an unsatisfied clause, WSAT is frequently "kicked out" to higher objective values—to the extent that it begins to spend significant time simply re-descending to a lower objective value, only to be prematurely kicked out again.

Although depth is a simplistic measure, it proves to be very useful for tuning noise and temperature parameters in local search procedures. By measuring depth, one can determine if the search is spending too much time recovering from large steps up in the objective and not enough time exploring near the bottom of the objective. More importantly, maintaining depth appears to be *necessary* for achieving reasonable search times. Figure 2 shows the results of a large experiment conducted over the entire collection of 2700 uf problems from SATLIB. This comparison ranked four comparable methods—SDF (introduced below), Novelty, Novelty+, and WSAT—in terms of their search depth and average flips. For each problem, the methods were ranked in terms of their average number of flips and average depth. Each (flips rank, depth rank) pair was then recorded in a table. The relative frequencies of these pairs is summarized in Figure 2. This figure shows that the highest ranked method in terms of search efficiency was always ranked near the best (and almost never in the bottom rank) in terms of search depth.

Although useful, depth alone is clearly not a *sufficient* criterion for ensuring good search performance. A local search could easily become stuck at a good objective value, and yet fail to explore widely. To account for this possibility we introduce another measure of local search effectiveness.



100 runs on uf100-0953	Avg. mobility	Avg. depth	Avg. flips	Est. opt. w/cutoff
GSAT$()$	5.7	2.13	99,006	52,100
HSAT$()$	7.1	2.10	99,006	12,300
WSAT-G$(.5)$	9.7	4.29	26,685	13,700
WSAT$(.5)$	15.7	5.65	11,342	9,421
Novelty$(.5)$	18.9	4.76	4,122	4,122
SDF$(\frac{2}{m}, .995)$	25.9	4.09	1,355	1,355

Figure 3: Mobility results

Mobility measures how rapidly a local search moves in the search space (while it tries to simultaneously stay deep in the objective). We measure mobility by calculating the Hamming distance between variable assignments that are k steps apart in the search sequence, and average this quantity over the entire sequence to obtain average distances at time lags $k = 1, 2, 3, ...$, etc. It is desirable to obtain a large value of mobility since this indicates that the search is moving rapidly through the space.

Mobility is obviously very important in a local search. In fact, most of the significant innovations in local search methods over the last decade appear to have the effect of substantially improving mobility without damaging depth. This is demonstrated clearly in Figure 3, again for the uf100-0953 problem. It appears that the dramatic improvements of these methods could have been predicted from their improved mobility scores (while maintaining comparable depth scores).

Figure 3 covers several highlights in the development of local search methods for satisfiability. For example, one of the first useful innovations over GSAT was to add a preference for least recently flipped variables, resulting in the superior HSAT procedure. Figure 3 shows that one benefit of this change is to increase mobility without damaging search depth, which clearly corresponds to improved solution times. Another early innovation was to incorporate "random walk" in GSAT. Figure 3 shows that WSAT-G also delivers a noticeable increase in mobility—again resulting in a dramatic reduction in solution times. It is interesting to note that the apparently subtle distinction between WSAT-G and WSAT in terms of their definition is no longer sub-

[1]The uf series of problems are randomly generated 3-CNF formulae that are generated at the phase transition ratio of 4.3 clauses to variables. Such formulae have roughly a 50% chance of being satisfiable, but uf contains only verified satisfiable instances. http://aida.intellektik.informatik.tu-darmstadt.de/∼hoos/SATLIB/

100 runs on uf100-0953	Avg. cover. rate	Avg. mob.	Avg. dep.	Avg. flips	Est. opt.
Novelty($.5$)	.0123	19	4.7	4,122	4,122
Novelty+($.5$)	.0227	19	4.9	2,830	2,298
SDF($\frac{2}{m}$, $.995$)	.0572	26	4.1	1,355	1,355

Figure 4: Coverage results

tle here: WSAT offers a dramatic improvement in mobility, along with an accompanying improvement in efficiency. Finally, the culmination of novelty and random walk in the Novelty procedure achieves even a further improvement in mobility, and, therefore it seems, solution time.

We have observed this effect consistently over the entire range of problems we have investigated. Thus it appears that, in addition to depth, mobility also is a necessary characteristic of an effective local search in SAT problems. To establish this further, Figure 2 shows the results of a large experiment on the entire collection of 2700 uf problems in SATLIB. The same four procedures were tested (SDF, Novelty, Novelty+, WSAT) and ranked in terms of their search mobility and solution time. The results show that the highest ranked in terms of mobility is almost always ranked near the top in problem solving efficiency, and that low mobility tends to correlate with inferior search efficiency.

A final characteristic of local search behavior that we consider is easily demonstrated by a simple observation: Hoos (1999) presents a simple satisfiable CNF formula with five variables and six clauses that causes Novelty to (sometimes) get stuck in a local basin of attraction that prevents it from solving an otherwise trivial problem. The significance of this example is that Novelty exhibits good depth and mobility in this case, and yet fails to solve what is otherwise an easy problem. This concern led Hoos to develop the slightly modified procedure Novelty+ in (Hoos 1999). The characteristic that Novelty is missing in this case is coverage.

Coverage measures how systematically the search explores the entire space. We compute a rate of coverage by first estimating the size of the largest "gap" in the search space (given by the maximum Hamming distance between any unexplored assignment and the nearest evaluated assignment) and measuring how rapidly the largest gap size is being reduced. In particular, we define the coverage rate to be $(n - \textit{max gap})/\textit{search steps}$. Note that it is desirable to have a high rate of coverage as this indicates that the search is systematically exploring new regions of the space as it proceeds.

Figure 4 shows that Hoos's modified Novelty+ procedure improves the coverage rate of Novelty on the uf100-0953 problem. Space limitations do not allow a full description, but Novelty+ demonstrates uniformly better coverage than Novelty while maintaining similar values on other measures, and thus achieves better performance on nearly every problem in the benchmark collections.

Overall, our results lead us to hypothesize that local search procedures work effectively because they descend quickly in the objective, persistently explore variable assignments with good objective values, and do so while moving rapidly through the search space and visiting very different variable assignments without returning to previously explored regions. That is, we surmise that good local search methods do not possess any special ability to predict whether a local basin of attraction contains a solution or not—rather they simply descend to promising regions and explore near the bottom of the objective as rapidly, broadly, and systematically as possible, until they stumble across a solution. Although this is a rather simplistic view, it seems supported by our data and moreover it has led to the development of a new local search technique. Our new procedure achieves good characteristics under these measures and, more importantly, exhibits good search performance in comparison to existing methods.

A new local search strategy: SDF

Although the previous measures provide useful diagnostic information about local search performance, the main contribution of this paper is a new local search procedure, which we call SDF for "smoothed descent and flood." Our procedure has two main components that distinguish it from previous approaches. First, we perform steepest descent in a more informative objective function than earlier methods. Second, we use multiplicative clause re-weighting to rapidly move out of local minima and efficiently travel to promising new regions of the search space.

Recall that the standard GSAT objective simply counts the number of unsatisfied clauses for a given variable assignment. We instead consider an objective that takes into account how many variables satisfy each clause. Here it will be more convenient to think of a reversed objective where we seek to maximize the number of satisfied clauses instead of minimize the number of unsatisfied clauses. Our enriched objective works by always favoring a variable assignment that satisfies more clauses, but all things being equal, favoring assignments that satisfy more clauses twice (subject to satisfying the same number of clauses once), and so on. In effect, we introduce a tie-breaking criterion that decides, when two assignments satisfy the same number of clauses, that we should prefer the assignment which satisfies more clauses on two distinct variables, and if the assignments are still tied, that we should prefer the assignment that satisfies more clauses on three distinct variables, etc. This tie-breaking scheme can be expressed in a scalar objective function that gives a large increment to the first satisfying variable, and then gives exponentially diminishing increments for subsequent satisfying variables for a given clause. For k-CNF formulas with m clauses, such a scoring function is

$$f_{\textit{ABE}}(\mathbf{x}) \quad = \quad \sum_c \textit{score}(\#\ x_i\text{'s that satisfy } c)$$

where $\textit{score}(0) = 0$, $\textit{score}(1) = m^{k-1}$, $\textit{score}(2) = m^{k-1} + m^{k-2}$,..., $\textit{score}(k) = m^{k-1} + m^{k-2} + \cdots + 1$.

Our intuition is that performing steepest ascent in this objective should help build robustness in the current variable assignment which the search can later exploit to satisfy new clauses. In fact, we observe this phenomenon in our experiments. Figure 5 shows that following a steepest ascent in

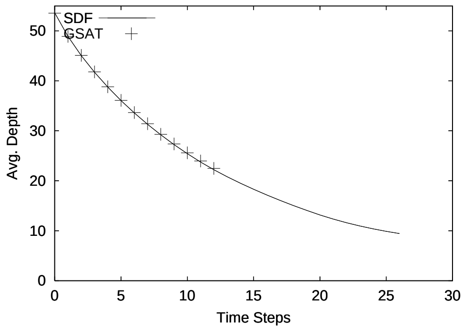


Figure 5: Descent results

f_{ABE} descends deeper in the original GSAT objective than the GSAT procedure itself (before either procedure reaches a local extrema or plateau). This happens because plateaus in the GSAT objective are not plateaus in f_{ABE}; in fact, such plateaus are usually opportunities to build up robustness in satisfied clauses which can be later exploited to satisfy new clauses. This effect is systematic and we have observed it in every problem we have examined. This gives our first evidence that the SDF procedure, by descending deeper in the GSAT objective, has the potential to improve the performance of existing local search methods.

The main outstanding issue is to cope with local maxima in the new objective. That is, although f_{ABE} does not contain many plateaus, the local search procedure now has to deal with legitimate (and numerous) local maxima in the search space. While this means that plateau walking is no longer a significant issue, it creates the difficulty of having to escape from true traps in the objective function. Our strategy for coping with local maxima involves the second main idea behind the SDF procedure: *multiplicative* clause re-weighting. Note that when a search is trapped at a local maxima the current variable assignment must leave some subset of the clauses unsatisfied. Many authors have observed that such local extrema can be "filled in" by increasing the weight of the unsatisfied clauses to create a new search direction that allows the procedure to escape (Wu & Wah 1999; Frank 1996; Morris 1993; Selman & Kautz 1993). However, previous published re-weighting schemes all use *additive* updates to increment the clause weights. Unfortunately, additive updates do not work very well on difficult search problems because the clauses develop large weight differences over time, and this causes the update mechanism to lose its ability to rapidly adapt the weight profile to new regions of the search space. Multiplicative updating has the advantage of maintaining the ability to swiftly change the weight profile whenever necessary.

One final issue we faced was that persistently satisfied clauses would often lose their weight to the extent that they would become frequently falsified, and consequently the depth of search (as measured in the GSAT objective) would deteriorate. To cope with this effect, we flattened the weight profile of the satisfied clauses at each re-weighting by shrinking them towards their common mean. This increased the weights of clauses without requiring them to be explicitly falsified and had the overall effect of restoring search depth and improving performance. The final SDF procedure we tested is summarized as follows.

SDF(δ, ρ) Flip the variable x_i that leads to the greatest increase in the weighted objective

$$f_{WABE}(\mathbf{x}) \;=\; \sum_c w(c)\ \mathit{score}(\#\ x_i\text{'s that satisfy } c)$$

If the current variable assignment is a local maximum and not a solution, then re-weight the clauses to create a new ascent direction and continue.

Re-weight(δ, ρ) Multiplicatively re-weight the unsatisfied clauses and re-normalize the clause weights so that the resulting largest difference in the f_{WABE} objective (when flipping any one variable) is δ. (That is, create a minimal greedy search direction.) Then flatten the weight profile of the satisfied clauses by shrinking them $1-\rho$ of the distance towards their common mean (to prevent the weights from becoming too small and causing clauses to be falsified gratuitously).

One interesting aspect of this procedure is that it is almost completely deterministic (given that ties are rare in the objective, without re-starts) and yet seems to perform very well in comparison to the best current methods, all of which are highly stochastic. We claim that much of the reason for this success is that SDF maintains good depth, mobility, and coverage in its search. This is clearly demonstrated in Figures 1–3 which show that SDF obtains superior measurements in every criteria.

Evaluation

We have conducted a preliminary evaluation of SDF on several thousand benchmark SAT problems from the SATLIB and DIMACS repositories. The early results appear to be very promising. Comparing SDF to the very effective Novelty+ and WSAT procedures, we find that SDF typically reduces the number of flips needed to find a solution over the best of Novelty+ and WSAT by a factor of two to four on random satisfiable CNF formulae (from the uf, flat, and aim collections), and by a factor of five to ten on nonrandom CNF formulae (from the SATLIB blocks-world and ais problem sets). These results are consistent across the vast majority of the problems we have investigated, and hold up even when considering the mean flips without restart, median flips, and optimal expected flips using restarts estimated from (1). However, our current implementation of SDF is not optimized and does not yet outperform current methods in terms of CPU time. Details are reported below.

In all experiments, each problem was executed 100 times and results averaged over these runs. All problems were tried with SDF$(\frac{.2}{m}, .995)$, Novelty$(.5)$, Novelty+$(.5)$, and WSAT$(.5)$. Furthermore, smaller problems were also tried with HSAT, GSAT, and simulated annealing.[2] There are

[2] We have as yet been unable to replicate the reported results for the DLM procedures from the published descriptions (Wu & Wah 1999; Shang & Wah 1998), and so did not include them in our study. This remains as future work.

	SDF		Novelty+		WSAT
	Avg.	Est. opt.	Avg.	Est. opt.	Est. opt.
uf50	214	140	411	271	505
uf75	488	411	978	732	1394
uf100	939	748	2297	1470	2877
uf125	1906	1563	5160	2712	5876
uf150	2962	2209	8253	4314	8393
uf175	6632	4945	18208	10719	20696
uf200	14106	9485	29536	18371	32039
uf225	17026	12198	38288	21129	35394
uf250	17980	13619	30617	23681	33993
flat125	15230	12868	37085	26369	64294
flat150	36020	31615	81668	59194	135981
aim100	94642	89155	236540	236215	236281

	SDF		Novelty+	% Failed
	Avg. flips	Est. opt.	Est. opt.	SDF / Nov+
bwlarge.a	2747	2701	10588	0 / 0
bwlarge.b	41907	39611	354111	0 / 40
bwlarge.c	470366	470366	-	87 / 100
huge	2561	2560	11104	0 / 0
logistics.c	17249	16870	140412	0 / 0

	SDF		WSAT	% Failed
	Avg. flips	Est. opt.	Est. opt.	SDF / WSAT
ais6	441	435	1063	0 / 0
ais8	6901	5617	34508	0 / 0
ais10	20214	16095	297460	0 / 28
ais12	154464	134491	488421	5 / 96

Figure 6: Search results

three sets of experiments reported in Figure 6. The first set covers a wide array of random SAT problems from both the SATLIB and DIMACS repositories. The results shown are averaged over all problems in the respective problem set and are shown for the runs with SDF, Novelty+ (which was 2nd best), and WSAT. The second set covers large planning problems. The results are shown for SDF and Novelty+ (2nd best), and the failure rates of each are compared. The third set covers the ais (All-Interval Series) problem set and shows results for SDF and WSAT (2nd best). In all experiments, the mean flips without restart and optimal expected flips are reported for SDF, and the optimal expected flips is reported for the other algorithms (when significantly smaller than the mean without restarts).

The results for the non-random blocks-world and ais problems are particularly striking. These problems challenge state of the art local search methods (verified in Figure 6) and yet SDF appears to solve them relatively quickly. This suggests that, although SDF shares many similarities to other local search methods currently in use, if might offer a qualitatively different approach that could yield benefits in real world problems.

The current implementation of SDF is unfortunately not without its limitations. We are presently using a non-optimized floating-point implementation, which means that even though SDF executes significantly fewer search steps (flips) to solve most problems, each search step is more expensive to compute. The overhead of our current implementation is about factor of six greater than that of Novelty or WSAT per flip, which means that in terms of CPU time, SDF is only competitive with the current best methods in some cases (e.g. bw_large.b). However, the inherent algorithmic complexity of each flip computation in SDF is no greater than that of GSAT, and we therefore expect that an optimized implementation in integer arithmetic will speed SDF up considerably—possibly to the extent that it strongly outperforms current methods in terms of CPU time as well.

References

Bayardo, R., and Schrag, R. 1997. Using CSP look-back techniques to solve real-world SAT instances. In *Proc. AAAI-97*.

Cha, B., and Iwama, K. 1995. Performance test of local search algorithms using new types of random CNF formulas. In *Proc. IJCAI-95*, 304–310.

Cha, B., and Iwama, K. 1996. Adding new clauses for faster local search. In *Proc. AAAI-96*, 332–337.

Frank, J. 1996. Weighting for Godot: Learning heuristics for GSAT. In *Proc. AAAI-96*, 338–343.

Frank, J. 1997. Learning short-tem weights for GSAT. In *Proc. IJCAI-97*, 384–391.

Gent, I., and Walsh, T. 1993. Towards an understanding of hill-climbing procedures for SAT. In *Proc. AAAI-93*, 28–33.

Gent, I., and Walsh, T. 1995. Unsatisfied variables in local search. In Hallam, J., ed., *Hybrid Problems, Hybrid Solutions (AISB-95)*.

Gomes, C.; Selman, B.; and Kautz, H. 1998. Boosting combinatorial search through randomization. In *Proc. AAAI-98*, 431–437.

Hoos, H. 1999. On the run-time behavior of stochastic local search algorithms for SAT. In *Proc. AAAI-99*, 661–666.

Kautz, H., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. AAAI-96*, 1194–1201.

Larrabee, T. 1992. Test pattern generation using boolean satisfiability. *IEEE Trans Computer-Aided Design* 11(1):4–15.

Li, C., and Anbulagan. 1997. Heuristics based on unit propagation for satisfiability problems. In *Proc. IJCAI-97*, 366–371.

Mazure, B.; Saïs, L.; and Grégoire, E. 1997. Tabu search for SAT. In *Proc. AAAI-97*, 281–285.

McAllester, D.; Selman, B.; and Kautz, H. 1997. Evidence for invariants in local search. In *Proc. AAAI-97*, 321–326.

Morris, P. 1993. The breakout method for escaping from local minima. In *Proc. AAAI-93*, 40–45.

Parkes, A., and Walser, J. 1996. Tuning local search for satisfiability testing. In *Proc. AAAI-96*, 356–362.

Selman, B., and Kautz, H. 1993. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proc. IJCAI-93*.

Selman, B.; Kautz, H.; and Cohen, B. 1994. Noise strategies for improving local search. In *Proc. AAAI-94*, 337–343.

Selman, B.; Kautz, H.; and McAllester, D. 1997. Ten challenges in propositional reasoning and search. In *Proc. IJCAI-97*, 50–54.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Proc. AAAI-92*.

Shang, Y., and Wah, W. 1998. A discrete Lagrangian based global search method for solving satisfiability problems. *J. Global Optimization* 12(1):61–99.

Wu, Z., and Wah, W. 1999. Trap escaping strategies in discrete Lagrangian methods for solving hard satisfiability and maximum satisfiability problems. In *Proc. AAAI-99*, 673–678.